REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 Sept 1997	Final 3/28/97 - 9/30/97

4. TITLE AND SUBTITLE
Trainer Methodology Study

5. FUNDING NUMBERS

6. AUTHOR(S)
Stephen J. Dow

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Alabama in Huntsville
Huntsville, AL 35899

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Aviation & Missile Command
Redstone Arsenal, AL 35898

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified, Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This task focused on development of two software programs for use with training devices which display moving targets overlaid on bitmapped terrain images. The first, called PathEdit, is used to define target paths across the terrain. The second, called DepthFinder, is used to produce per-pixel depth data for terrain images generated by taking photos with a digital camera. The report describes and documents these two programs.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Computer graphics, terrain imagery	27
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	None

PLEASE CHECK THE APPROPRIATE BLOCK BELOW

DAO# _____

☐ _____ copies are being forwarded. Indicate whether Statement A, B, C, D, E, F, or X applies.

☑ DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

☐ DISTRIBUTION STATEMENT B:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT C:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTS (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT D:
DISTRIBUTION AUTHORIZED TO DoD AND U.S. DoD CONTRACTORS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT E:
DISTRIBUTION AUTHORIZED TO DoD COMPONENTS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT F:
FUTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date) or HIGHER DoD AUTHORITY.

☐ DISTRIBUTION STATEMENT X:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DoD DIRECTIVE 5230.25. WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE, 6 Nov 1984 (indicate date of determination). CONTROLLING DoD OFFICE IS (Indicate Controlling DoD Office).

☐ This document was previously forwarded to DTIC on _____ (date) and the AD number is _____.

☐ In accordance with provisions of DoD instructions. The document requested is not supplied because:

☐ It will be published at a later date. (Enter approximate date, if known).

☐ Other. (Give Reason)

DoD Directive 5230.24, "Distribution Statements on Technical Documents," 18 Mar 87, contains seven distribution statements, as described briefly above. Technical Documents must be assigned distribution statements.

Stephen J. Dow
Print or Type Name

Stephen J Dow
Authorized Signature/Date

256 - 890 - 6406
Telephone Number

Trainer Methodology Study

Final Report

Principal Investigator and Author:	Stephen J. Dow, Department of Mathematical S\ The University of Alabama in H .lle

Date:	September 30, 1997
Contract Name:	F/DOD/ARMY/MICOM/Trainer Methodology Study
Contract Number:	DAAH01-91-D-R005 D.O. 75

1. Introduction

As specified in the Scope of Work, this task focused on development of two software programs for use with training devices which display moving targets overlaid on bitmapped terrain images. The first part of the task was to develop a program called PathEdit used to define the paths which the targets follow across the terrain. The second part of the task was to develop a program called Depth Finder used to produce per-pixel depth data for terrain images generated by taking photos with a digital camera. The two executable programs together with their source code are being delivered on floppy disk as part of this task. This report describes and documents the two programs.

In a number of places this report refers to the "existing BST system." This phrase refers to a training device currently in use called the Basic Skills Trainer. The BST uses custom hardware to perform the graphics operations for panning the terrain image and animating the targets. The programs developed for this task serve the dual purposes of (1) generating new data for use within the existing BST system and (2) providing proof of concept and software components for a replacement to the existing BST requiring only standard PC hardware to generate the training scenario animations.

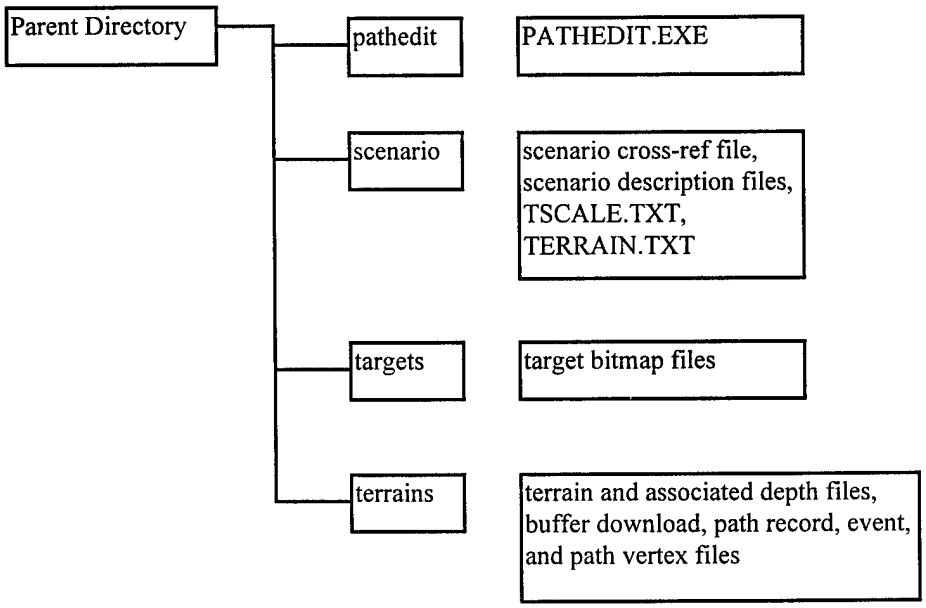
Both PathEdit and Depth Finder use the Win32 API, and so are designed to run on a personal computer running either Windows 95 or Windows NT. PathEdit can be run with the display configured for 256 or more colors. Depth Finder, which is designed to work with 24-bit-per-pixel images , requires the display to be configured for "high color" (16-bit, 65536 colors) or "true color" (24-bit, 16 million colors).

In this report technical terms requiring definition are underlined when being introduced and defined.

2. The PathEdit Program

2.1 Overview and File Structure

PathEdit is designed to generate scenario description, path record, and event files which conform to the format used by the existing BST (Basic Skills Trainer). Those file formats are documented elsewhere, so some details will be omitted here. The following directory structure is assumed by the PathEdit program:



The parent directory can be the root directory or any subdirectory and its name is not important. However, it must have three subdirectories named "scenario", "targets", and "terrains" and the PathEdit executable file (PATHEDIT.EXE) must reside in a subdirectory (whose name does not actually need to be "pathedit") at the same level as these other three.

A terrain image is stored as an 8-bit-per-pixel bitmap (.BMP) file containing a color palette. The terrain name is the string of up to 8 characters which is the part of the filename preceding the period; for example, the terrain named "BUILTUP" is stored in file BUILTUP.BMP. Each terrain has an associated depth file with filename consisting of the terrain name followed by the extension DEP; for example BUILTUP.DEP. This file contains an 8-bit depth value for each pixel in the terrain file, and nothing else (no header or other formatting bytes). Each terrain also has a two-digit terrain id number, used to designate the terrain in other files and filenames. The list of terrains available within PathEdit is specified

by TERRAIN.TXT, which is an ASCII text file containing one line for each terrain, listing the terrain id, terrain name, and three numbers which are viewing parameters explained in section 2.2.

Targets are drawn over the terrain in one of 48 views depending on which way the target is oriented relative to the viewer. This is accomplished by storing the 48 views as separate bitmaps within an IND file. Each target type has a name consisting of up to eight characters used to name the file. For example, the 48 views of the T72 target are stored in file T72.IND in the targets directory. The IND file is simply the concatenation of the 48 bitmap files holding the different views. The list of target types available within PathEdit is specified by TSCALE.TXT, which is an ASCII file containing one line for each target type, listing the target name, a scale factor, and optionally the letter 'F' designating the target as a flying target. In drawing the target over the terrain image, the target is scaled based on a range value for the current target location and the scale factor found in TSCALE.TXT and then two clipping conditions restrict which pixels are drawn. The first condition is that only nonzero-valued pixels in the target bitmap are drawn; thus zero is treated as the background or transparent color when the target bitmaps are created. The other condition is a depth comparison made between a single depth value for the current target location and each pixel in the (.DEP) depth file described above, thus allowing the whole target or a portion of it to be occluded by objects deemed closer. The method of obtaining determining a range and a depth value for a target location is discussed below.

In the terminology of the BST, a scenario consists of a choice of terrain together with a set of targets, each target having a designated target type and target path. A target path specifies the locations visited by the target and timing information for moving from location to location. Each scenario has an associated four-digit scenario id number.

The scenario cross-reference file is an ASCII text file named SCEN_TER.XRF containing one line for each available scenario, listing the scenario id number, terrain id number, and terrain name for that scenario. Each scenario has an associated scenario description file named SCENXXYYYY.DES, where XX is the terrain id and YYYY is the scenario id. The scenario desciption file is also an ASCII text file, and contains a single header line followed by one line for each target path in the scenario, listing the path id, starting frame number, target name, target color, and a flag indicating whether the target is moving or stationary. The path id is a four-digit number used to specify the other files describing the path. The starting frame number is always 1. A frame as used here is the basic clock unit used in defining paths, and equals 1/60 of a second.

The files associated with a path are the path record file (.REC), event file (.EVT), buffer download file (.BDC), and path vertex file (.PTH). The first three file types are required by the existing BST system, whereas the path vertex files are specific to PathEdit. The path record file is a binary file containing records for each pixel location visited by a target following the path. The event file is a binary file containing records indicating points along the path where the status of the target changes. The buffer

download file is a binary file associated with a specific target path in a specific scenario, indicating the assignment of image buffers within the BST for that target. Buffer download files are generated by PathEdit but not used within PathEdit. Path vertex files are used by PathEdit in editting paths; these files are ASCII text files containing one line for each vertex along a path, a vertex being a location where the target changes speed or direction. Between vertices a target moves in a straight line at constant ground speed, or is stationary. The portion of a path joining two vertices is referred to as a path segment. Vertex locations are given in a 3D coordinate system defined with respect to the ground and viewing location. We will call these 3D coordinates ground coordinates; the 2D coordinates specifying a column and row number in the terrain image will be referred to as pixel coordinates. Section 2.2 explains the math model used to relate ground and pixel coordinates. Each line in a path vertex file contains five fields: the first three giving the x, y, and z ground coordinates of the vertex location, the last two giving the frame count and view number for the segment joining the current vertex and the one given on the following line. The frame count and view number on the last line of the file refer to an implicit stationary segment at the end of the path; i.e. once the target reaches the vertex location given in the last line of the file it remains at that location for the number of frames given in the last line. The view number given in the file is in the range 0 to 47; whereas it is displayed to the user running PathEdit as a number in the range 1 to 48.

As mentioned above, each pixel location visited by a target along its path is stored in a path record. In addition to two fields for the x and y pixel coordinates, a path record contains a frame count, target range, depth, buffer number, and rotation value. PathEdit generates these path records from the path vertices and associated information. We now give an overview of how this is done.

Each path segment joining two vertices is subdivided by PathEdit into path records containing each of the pixel locations along that segment This subdivision splits up the frame count for the path segment into frame counts for the path records which add up to the specified segment total. PathEdit does this subdivision using the ground coordinates of the vertex endpoints so that the frame counts of the path records reflect a target moving at constant ground speed along the segment. Ground coordinates along the segment are converted to pixel coordinates to go into the path records. The range value is taken to be equivalent to the ground z-coordinate as discussed in section 2.2. PathEdit does not deal with the path record rotation values; it always generates zero for that field. This leaves the depth and buffer numbers to be explained.

The depth value stored in the path record is determined by reference to a table relating pixel y-coordinates to depths. One such table is stored for each terrain in the form of an ASCII file DEPTHXX.TXT, where XX is a terrain id, in the terrains directory. The file lists on each line a raster line number number followed by a integer depth value between 0 and 255. The depth value indicates the depth of an object on the ground and displayed on the given raster line. Thus to generate the path record depth field for a target on the ground, PathEdit simply looks up the value in this table using the y-pixel

coordinate already obtained for the path record. For a flying target, PathEdit does the same lookup, except that the y-pixel coordinate is obtained by converting to pixels the point beneath the target on the ground.

The buffer number in a path record specifies a particular view of a particular target type through a lookup scheme involving the assignment of these views to buffers given in the buffer download files. Details of that lookup scheme are not detailed here because it is specific to the existing BST system and documented there; however it should be noted that the scheme requires the REC and BDC files to be generated together for the whole scenario, since they contain indices which are meaningful only within that context. The PTH files are not restricted in that way, but rather are essentially tied only to a particular terrain.

2.2 Math Model

This section explains the modeling of the imaging process used to relate 3d ground coordinates to pixel coordinates of the terrain files.

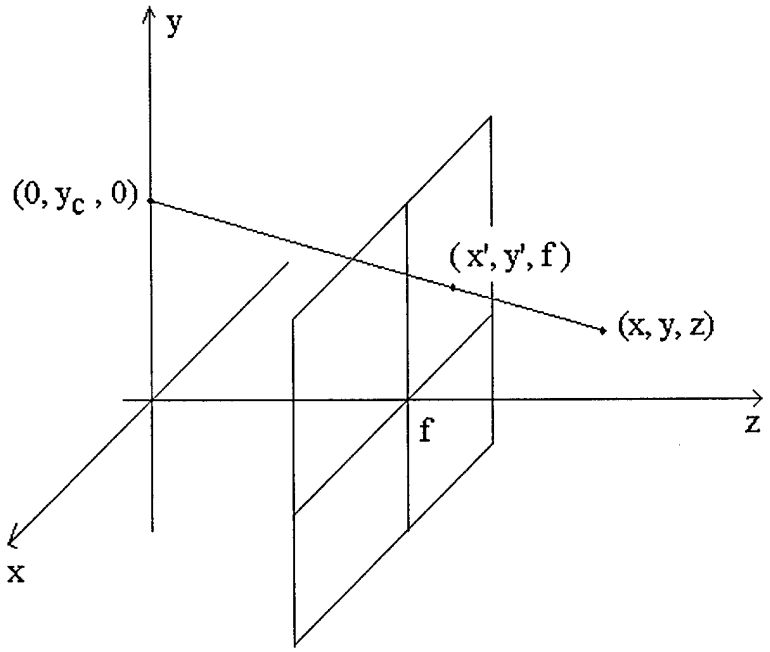


Figure 1: Math Model used by PathEdit

As illustrated in the diagram above, the model being used is simple perspective projection, which projects a point in space onto an image plane by intersecting that plane and the straight line joining the point and the center of projection. The center of projection, which can be thought of as the camera location, is located on the y-axis at $(0, y_c, 0)$ and the image plane is the plane $z = f$. The viewing direction is positive z. Labeling an arbitrary point in space (x, y, z) and its projection in the image plane (x', y', f) as in the diagram, collinearity of these two points and the center of projection implies:

$$(x', y', f) = [(x, y, z) - (0, y_c, 0)]\ f/z, \text{ or}$$

$$x' = xf/z$$

$$y' = (y - y_c)f/z$$

Bitmap image coordinates (x", y") are obtained from (x', y') by a scale and translation:

$$x'' = sx' + x_0 = x(sf)/z + x_0$$

$$y'' = -sy' + y_0 = -(y - y_c)(sf)/z + y_0$$

The original coordinates (x, y, z) are in meters, whereas (x", y") are in pixels (the pixels of the image file; further scaling may be applied in generating screen coordinates when "zooming" in or out). We use the convention that image file coordinates start at (0, 0) in the upper left hand corner; hence the negative sign in the equation for y" above. Because parameters s and f appear in the equations above only together as a product sf, we regard them as a single parameter sf to be determined.

Additional assumptions made within PathEdit are that the ground x-coordinate at the center of the image is zero, which implies that $x_0 = (xsize - 1)/2$, where xsize is the size in pixels of the image in the x direction, and that the ground is the xz-plane, which implies that y_0 is the image coordinate of the horizon. Parameters y_0, y_c, and sf are the three viewing parameters mentioned in section 2.1 as being stored for each terrain in the TERRAIN.TXT file. PathEdit has a menu option ("Edit Terrain Parameters") for editting the values of these three parameters. As mentioned above, y_0 should be entered as the y-pixel coordinate of the horizon line; for example $y_0 = 108$. The other two values can be determined from the desired ranges and scaling of objects in the terrain image, using the fact that z-coordinates correspond to range and sf/z represents the meter-to-pixel scaling factor for objects with a given z-coordinate. For example, for BUILTUP.BMP we may decide that one of buildings' doors, located at y" = 300 and 15 pixels high, should be at a range of z = 420 and be 2.5 meters high. Then sf and y_c may be determined as follows:

$$sf/420 = 15/2.5, \text{ or } sf = 2520.$$

$$300 = (2520/420)y_c + 108, \text{ or } y_c = 32.0$$

2.3 Program Operation

2.3.1 Initial Windows

The initial windows displayed by PathEdit on startup are shown in Figure 2. On the left is the main program window, which serves to display the terrain image and overlaid targets and active target path. It also has the main program menu at the top and a "Run Info" display attached along the bottom edge. Closing this window exits the program. On the right is the "Edit Scenario" dialog box. On startup

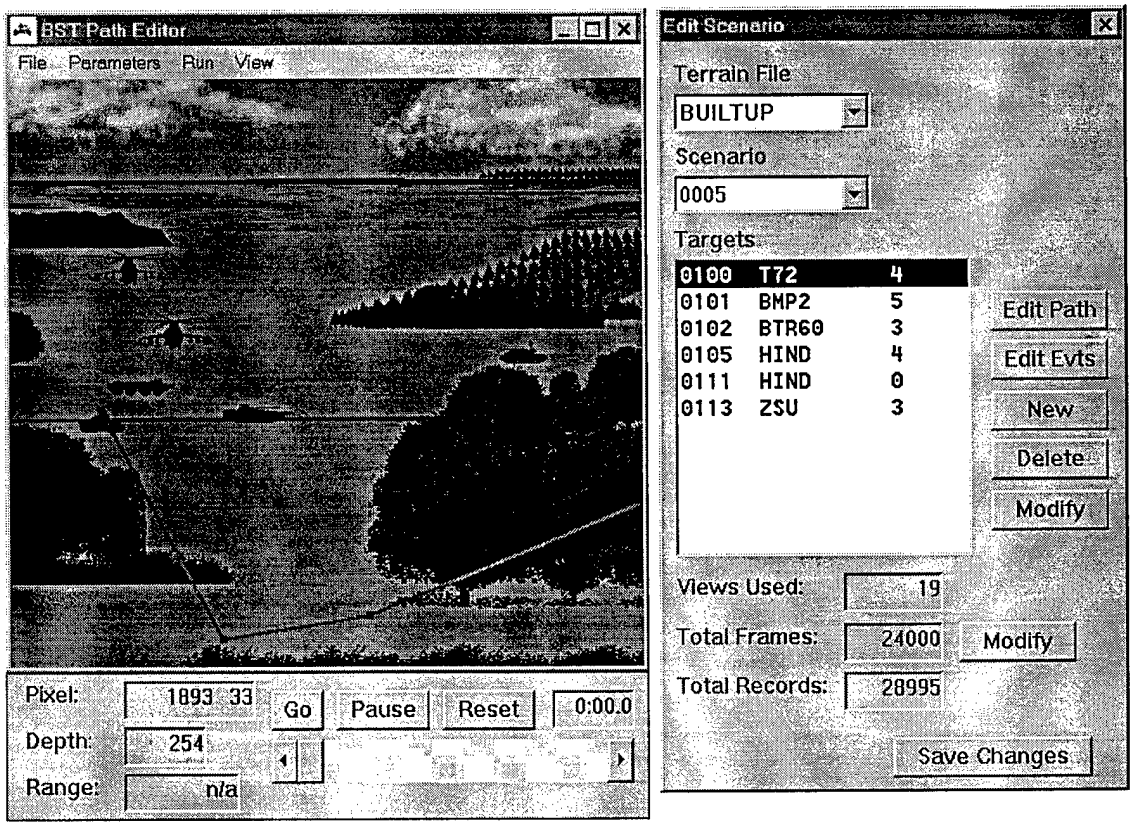


Figure 2: Initial PathEdit Windows

the main window will be blank, indicating no terrain file or scenario has been loaded; this will also be reflected by the top two fields of the Edit Scenario dialog box being blank. A terrain may be loaded by selecting from the drop-down terrain list, or a scenario loaded by selecting from the drop-down scenario list. Loading an existing scenario automatically loads its associated terrain. To create a new scenario, first select a terrain, then select the last entry (labeled "new") in the scenario list.

Further discussion of the edit scenario dialog box is given in the next section. We now briefly describe the controls on the "Run Info" display at the bottom of the main program window. At the left are

fields for pixel coordinates, depth, and range. These fields are readouts which update as the mouse cursor moves over the terrain image displayed above. The range displayed assumes that the mouse cursor is on the ground; when it moves to a location above the horizon the readout changes to n/a. All of these readouts also change when the active target moves as a result of user interaction on the edit path and edit events dialog boxes. In that situation, the readouts typically show the parameters for the active vertex, as discussed below in regard to the edit path dialog box. However, whenever the mouse cursor is moved over the terrain display, the readouts will again update to reflect its location.

The controls to the right are the current frame scrollbar and associated go, pause, and reset buttons, and timer readout. These control and display information about the scenario timeline. Whenever a scenario is loaded within PathEdit, the user can run or pause the scenario, using the go and pause buttons or corresponding options on the run menu. The reset button returns the scenario to the initial frame. The scrollbar extents correspond to the length of the scenario, so that dragging the scrollbar button to the left or right end of the scrollbar takes the user to the initial or final frame of the scenario. The timer readout shows the current position along the timeline in tenths of a second.

2.3.2 Editting a Scenario

Once a scenario has been loaded, the list of targets for that scenario will be displayed in the dialog box list as shown in Figure 2. Each line in the target list shows a path id, a target name, and a number representing the number of distinct views of the target needed for that path. This may be less than the number of segments in the path, because multiple segments may use the same view and also because views which are accounted for by preceding targets in the list of the same target type are not counted again. Thus the numbers in the right column add up to the number in the field below labeled "Views used," which represents the overall number of distinct target views used by the scenario. This display is useful because the existing BST system has an overall limit of 31 target views per scenario.

We will refer to the target which is currently highlighted in the target list as the active target. The path of the active target is drawn in the main display window.

The field labeled "Total Frames" shows the running time of the scenario in frames (sixtieths of a second). A button beside the field brings up a dialog box which allows the user to modify the total frames value. That dialog box shows a breakdown of the total frames into "frames with movement" and "trailing static frames." Recall from the discussion in Section 2.1 that each path is defined by a sequence of vertices, each of which has an associated frame count. For all the vertices except the last this frame count refers to the time spent between that vertex and the following one. The frame count for the last vertex is time spent sitting at that location while the total frame count for the scenario runs out. While editting paths as discussed below, PathEdit adjusts this last frame count for each path in the scenario so that the total frame count for each path matches the total frames for the scenario. When modifying the total frame count for the scenario, the user is not allowed to shorten the frame count for any path already in the scenario,

except by reducing the frame count of the path's last vertex. To further reduce the total frames for the scenario, the user must remove the target with the long path from the scenario or edit the long path, reducing its frames with movement, and then return to the modify total frames dialog box.

The field labeled "Total Records" displays the total number of path records for the scenario; i.e. the sum of the lengths (in records) of the paths in the scenario. This is helpful information because the existing BST has a limit on this number.

The buttons to the right of the target list on the edit scenario dialog are:

(1) Edit Path	This button closes the edit scenario dialog box and brings up a dialog box for editting the path of the active target.
(2) Edit Evts	This button closes the edit scenario dialog box and brings up a dialog box for editting the events along the path of the active target.
(3) New	This button adds a new target to the scenario. A dialog box comes up allowing the user to select the target type and color and to select either an existing path for the new target or to create a new path. If the user chooses to create a new path the user will be taken directly into the edit path dialog box.
(4) Delete	This button removes the active target from the scenario. This action does not delete the target bitmap or other files associated with the target; it just deletes the entry for the target in the scenario description file.
(5) Modify	This button brings up a dialog allowing the user to change the target type and/or color for the active target.

2.3.3 Editting a Path

Figure 3 shows the edit path dialog box. Several user actions cause this dialog box to come up: hitting the edit path button on either the edit scenario dialog box or the edit events dialog box, selecting edit path on the file menu, or choosing to create a new path when adding a new target to a scenario. A list near the top of the dialog box displays the path vertices with fields labeled frame, x, y, view, and frame_ct; these fields hold the starting frame number for the vertex, x and y pixel coordinates, view number (1-48), and frame count. As mentioned earlier, the frame count for a vertex represents the number of frames used in moving from the given vertex to the next vertex, except for the last vertex where it represents the number of frames for which the target remains at the last vertex location while the scenario runs to completion. Thus the values in the frame_ct column add up to the total frames for the scenario. As the path is editted, the last vertex frame count is adjusted to maintain that sum, so the last frame count may be regarded as a reserve which the user is drawing down. If the user edits the path in such a way as to cause this reserve to run out, so that the number of frames of movement exceeds the scenario total frames, then the scenario total frames value is increased and a warning message box appears to tell the user that this has

happened. In this situation closing the edit path dialog box without saving the results will cause the scenario total frames to be reset to its previous value.

The active target path is displayed in the main program window, with small boxes indicating the vertices. The vertex corresponding to the highlighted row in the vertex list is called the active vertex. A vertex may be made active by selecting it in the vertex list or by moving the mouse cursor near one of the vertex boxes. Making a vertex active causes the scenario to move to the starting frame number for that vertex (and pauses the scenario at that point if it was running). Information about the active vertex and the associated path segment is displayed in the fields below the vertex list, and most of the editting options available to the user apply to this vertex or segment. The user may type values into the pixel location or ground speed fields (ground speed is in meters/sec) and hit the apply button to change these parameters. A vertex is called a stationary vertex if the next vertex in the list is at the same location or if it is the last vertex. When a stationary vertex becomes active, the ground speed field changes to a frame count field, allowing the user to enter the number of frames at which the target is stationary at that location. The view number for a vertex may be changed using the up/down arrows or computed based on the direction of the path segment using the compute button. These changes take place immediately and the display of the target updates to reflect the change. Certain target types are designated as being flying targets. When the active target is a flying target, the edit path dialog box displays an altitude field in place of the view number field.

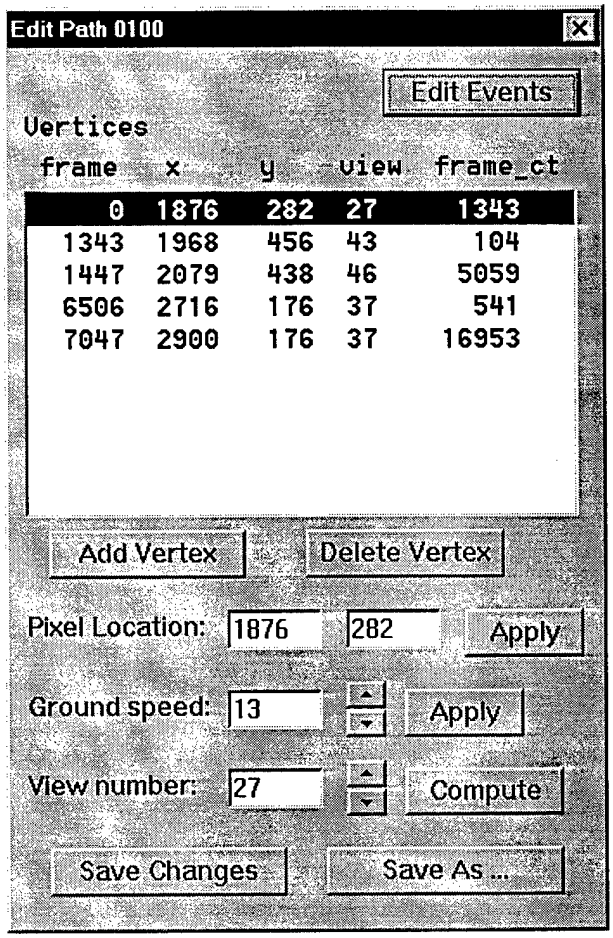


Figure 3: Edit Path Dialog Box

The add and delete vertex buttons are fairly self-explanatory. When adding a vertex, the new vertex goes after the active vertex and becomes the new active vertex. The user has the choice of adding the new vertex at the same location or at a new location. In the first case, the previously active vertex becomes a stationary vertex. In the second case the new vertex is added at the halfway point along the active segment, or if it is being added at the end of the path, at a location to the right of the last vertex.

The active vertex may be moved interactively with the mouse, in the standard manner of drag/drop mouse operations using the left mouse button. When this is done with a vertex where one or

more stationary vertices are located, the whole set of vertices at that location move together. Frame counts are adjusted for the preceding and/or following segments so as to maintain the ground speeds along those segments at their previous values. This means that moving a vertex typically draws down or adds to the frame count for the last vertex. Other editting operations such as changing a ground speed or in some cases adding or deleting a vertex have this effect as well.

When the path has been editted as desired, the user may save the changes to the current path vertex file (which is determined by the path id number) using the save button. The path may be saved to a new file based on a new path id using the "save as" button. Note: saving changes on either the edit path dialog box or the edit events dialog box actually saves both the path vertex (PTH) file and the event (EVT) file for the path, but not the buffer download and path record files. The latter files tie the whole scenario together and therefore are only saved at the time the scenario description file is saved, by using the save button on the edit scenario dialog box.

2.3.4 Editting Events

Associated with each path is a sequence of events, which record changes to the status of a target as it moves along its path. Figure 4 shows the dialog box used to edit these events for the active target path. The target status consists of 4 states, namely a clutter state, attack state, solution state, and hitable state. Each of these states is one of an enumerated list of possible values for that state; for example the possible clutter states are "open", "medium", "maximum", and "total." These state values and their meanings are given in the documention of the existing BST system and so will not be detailed further here.

As a target moves along its path, one or more of its states may change. Such a change is called an event. Events are listed on the dialog box by showing on the first line the initial value for all four states and on succeeding lines displaying only

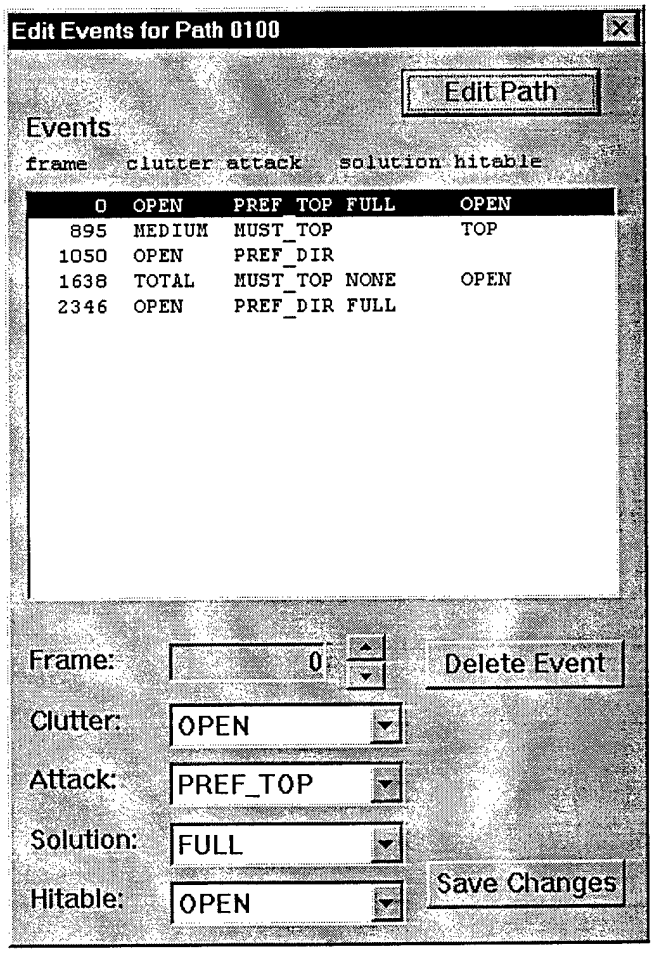


Figure 4: Edit Events Dialog Box

those values which change from the one above. Event locations are displayed in the main display window as small yellow triangles drawn on the active path. Selecting a line in the event list causes the scenario to move to the frame number at which the corresponding event occurs, displaying that frame number in the field below along with the values of the four states in the further fields below. The frame location of the event may be adjusted forward or back using the arrow buttons beside the frame number field. The values of the state variables may be changed using the drop-down lists on the state fields. The highlighted event may be deleted with the delete button. An new event is inserted by moving to the frame where the event is to occur (using the scrollbar at the bottom of the main display window) and then changing one of more of the state fields on the edit events dialog box.

A button is provided to save changes to the event list. As noted earlier, saving changes on either the edit path dialog box or the edit events dialog box actually saves both the path vertex (PTH) file and the event (EVT) file for the path.

2.3.5 Additional Menu Items

The remaining items on the PathEdit menus will now be discussed. The items on the File menu are Edit Scenario, Edit Path, Edit Events, Close Scenario, and About Pathedit. The first three of these are used to move to one or the other of the three corresponding edit dialog boxes discussed in the previous sections. Close Scenario is used to return to the state where only a terrain but not a scenario is loaded. This is needed because the first two menu items under the Parameters menu are available only in that state.

The items on the Parameters menu are Edit Terrain Parameters, Edit Depth Table, and Display Ranges. Edit Terrain Parameters brings up a dialog box allowing the three viewing parameters discussed in the section 2.2 to be changed. Edit Depth Table brings up a dialog box allowing the user to generate or edit the DEPTHXX.TXT file discussed in section 2.1. That file contains a table relating raster line numbers and ground depth values. While the dialog box is displayed, depth values which have been loaded from the existing file are shown only in a column labeled "old" and a column labeled "new" is blank. Pressing the left mouse button down while moving the mouse cursor over ground locations in the image fills values into the "new" list from those found at the pixel locations of the mouse in the terrain depth (DEP) file. The user can then cause those new values to replace the old in the DEPTHXX.TXT file by hitting the save changes button.

The Display Ranges menu item is available only when a scenario is loaded. It brings up a dialog box listing the minimum and maximum ranges occurring within each path in the scenario. This dialog only displays information and cannot be used to edit or change any parameters of the scenario. The refresh button causes the values displayed to come up to date with any changes caused by edits which have occurred since the dialog box was displayed.

The items on the Run menu are Test Mode, Go, Pause, Reset, and Display Run Info. The Go, Pause, and Reset items are menu equivalents of the corresponding buttons displayed at the bottom of the

main program window. Test Mode refers to running the scenario outside the context of any editting. When in test mode, the available scenarios are listed at the bottom of the file menu, and these include scenarios for which no path vertex files are available. Also test mode allows the mouse to be used to pan around the terrain image by clicking on the left mouse button within the display window and then moving the mouse. To return from this panning mode, click the left mouse button again. To go back to the normal editting mode, select edit scenario from the file menu. The Display Run Info only works in test mode and toggles display of the run controls at the bottom of the display window.

The items on the View menu are Zoom In, Zoom Out, AutoScroll, Draw Active Path, and Repaint. Zoom In, Zoom Out, and Repaint are fairly self-explanatory. The other two items are toggles, which when turned on show a check mark. When AutoScroll is on, moving along the scenario timeline causes the display to scroll as needed to keep the active target in view. The Draw Active Path setting determines whether the active target path is drawn. Normally one would want it turned on while editting, but perhaps turned off to view the results.

3. The Depth Finder Program

3.1 Overview

The second problem addressed during this task is that of generating per-pixel depth information for digital photographs of real terrain. We note here at the outset that whereas the PathEdit program is in a fairly mature state, at least for the problem of creating and editting paths for the artificial terrain scenes used at present, the Depth Finder program should be considered a work in progress or demonstration program at this stage of its development. A follow-on task to continue developing this program is already in place. However the program as developed to this point does demonstrate that the approach to extracting depth information works and, with sufficient time and effort on the user's part, the program can be used to generate per-pixel depth images for real-world terrain using photography from an inexpensive digital camera. This study was carried out working with a midrange consumer level camera, the Kodak model DC50.

Depths are derived from a pair of photographs of the terrain from slightly different vantage points; called a stereo pair. Figure 5 shows the display of a stereo pair within Depth Finder. Extracting 3D information from stereo photography is a standard problem (perhaps the central focus) of photogrammetry. Methods of solving this problem are discussed in a number of textbooks as well as the encyclopedic *Manual of Photogrammetry* published by the American Society of Photogrammetry. While our problem is a bit different from the most common case dealt with in the realm of photogrammetry (vertical aerial photographs taken with metric cameras), the basic techniques are the same. As described in more detail in the following sections, we first derive interior orientation (IO) parameters allowing pixel coordinates in the digital images to be related to real-world coordinates on a virtual imaging plane; then object or feature locations appearing in both photos are used to derive the relative orientation (RO) of the two cameras to one another. Together the parameters of IO and RO allow one to derive 3D coordinates of points relative to the (left) camera, called model coordinates, by finding the intersection of reconstructed rays from the two cameras back to the object. The z-component of these model coordinates is the point's depth.

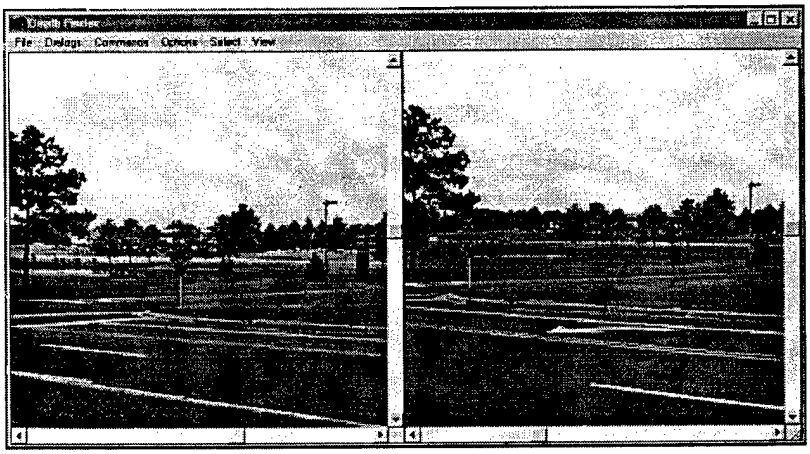


Figure 5: Displaying a Stereo Pair in Depth Finder

So far this explains how to obtain depths for individual points which have been measured on both photos. Here measuring a point refers to identifying the pixel coordinates of the point. At present this measurement is done by manually placing the mouse cursor at the point's location in the image display window and clicking a mouse button, but work is currently in progress to add computer assistance and automation to the point measurement process. Depth Finder displays a small cross symbol at each measured point. Figure 5 shows a number of these (red) crosses; for example there are several points around the perimeter of the stop sign which have been measured on both photos.

Having obtained depths for individually measured points, the other major step to be performed is to fill in depth values for the various regions of the image between and around those points. The left image is used as the basis for this process, or more precisely the portion of the left image representing terrain also visible in the right image, called the overlap area. The goal is to assign a depth value to each pixel in this overlap area. Again development of computer assistance and automation techniques for this problem is underway, while at present the method of filling in these depths is a mostly manual operation, based on the methodology of computer "paint" programs. The user defines a region of pixels, called the current selection, and then applies either a single depth or a planar depth equation to this set of pixels. In the first case the depth of one of the measured points is used or a manually entered value. In the second case, several of the measured points are used to define the planar surface, such as several points on a relatively flat portion of the ground or an object such as the stop sign of Figure 5.

Each of the two sets of parameters (IO and RO) is obtained by the method of least squares adjustment, a mathematical procedure discussed in general terms in section 3.2. Then the specifics for each of the two adjustments are discussed in sections 3.3 and 3.4. Section 3.5 describes the files used by Depth Finder to hold parameter values, point measurements, depth images and user preferences. Finally section 3.6 briefly describes operation of the Depth Finder program itself.

3.2 Mathematical Preliminaries: Least Squares Adjustments

This section contains background material on the mathematical procedure known as least squares adjustment which underlies the methods in the next two sections. This section may be omitted on a first reading or by those not needing to follow all the mathematical details.

The term adjustment in this context derives from the idea that measurements contain small errors which render them inconsistent with equations (conditions) which they should in theory satisfy; thus the measurements need to be "adjusted." However for our purposes the usefulness of the adjustment procedure is not to obtain better measurements but rather to determine the values of parameters involved in the condition equations. The general formulation we will be using starts with t condition equations relating n unknown parameters $x_1, x_2, ..., x_n$ and q measurement variables $y_1, y_2, ..., y_q$. The measurement data available for the adjustment consists of k vectors, each specifying values for $y_1, y_2, ..., y_q$. The condition equations may be written as

$$f_1(x_1, x_2, ..., x_n, y_1, y_2, ..., y_q) = 0$$
$$f_2(x_1, x_2, ..., x_n, y_1, y_2, ..., y_q) = 0$$
$$\cdots$$
$$f_t(x_1, x_2, ..., x_n, y_1, y_2, ..., y_q) = 0$$

The set of t equations may be expressed as a single vector equation $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$. For the moment we suppress the dependence on the measurements and treat $\mathbf{f}$ as a function of $\mathbf{x}$ only. Let $\mathbf{x}_0$ denote an initial approximation for $\mathbf{x}$. Assuming that $\mathbf{f}$ is differentiable at $\mathbf{x}_0$, we may write $\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + d\mathbf{f}(\mathbf{x}_0)\Delta\mathbf{x} + \varepsilon(\Delta\mathbf{x})$, where $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$ and ε is the error function representing the deviation of $\mathbf{f}$ from its linear approximation; it satisfies $\varepsilon(\Delta\mathbf{x})/\Delta\mathbf{x} \to 0$ as $\Delta\mathbf{x} \to 0$. The differential $d\mathbf{f}(\mathbf{x}_0)$ is the matrix of partial derivatives evaluated at $\mathbf{x}_0$:

$$d\mathbf{f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_t}{\partial x_1} & \cdots & \frac{\partial f_t}{\partial x_n} \end{bmatrix}$$

If the initial approximation is good, we may improve on it by ignoring the error term and solving the equation

$$\mathbf{f}(\mathbf{x}_0) + d\mathbf{f}(\mathbf{x}_0)\Delta\mathbf{x} = \mathbf{f}(\mathbf{x}) = \mathbf{0}, \text{ or}$$
$$d\mathbf{f}(\mathbf{x}_0)\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}_0).$$

This last equation is a vector equation having t components. Thus it is a system of t linear equations in n unknowns whose coefficients depend on the measurements. By plugging in each of the k measurement vectors $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k$, we get m = tk linear equations in n unknowns: $B\Delta\mathbf{x} = C$, where $B = [\, d\mathbf{f}(\mathbf{x}_0, \mathbf{y}_1)\ d\mathbf{f}(\mathbf{x}_0, \mathbf{y}_2)\ \ldots\ d\mathbf{f}(\mathbf{x}_0, \mathbf{y}_q)\,]^T$ and $C = [\, -\mathbf{f}(\mathbf{x}_0, \mathbf{y}_1)\ -\mathbf{f}(\mathbf{x}_0, \mathbf{y}_2)\ \ldots\ -\mathbf{f}(\mathbf{x}_0, \mathbf{y}_q)\,]^T$. If there are sufficient measurements, the system of equations is overdetermined. Its least squares solution is given by $\Delta\mathbf{x} = (B^TB)^{-1}B^TC$. The vector $\Delta\mathbf{x}$ thus obtained is added to the initial approximation $\mathbf{x}_0$ to obtain the new approximation $\mathbf{x}$. This process is repeated until a convergence criterion is met, such as that all components of $\Delta\mathbf{x}$ are smaller than some predefined value.

3.3 Camera Geometry and Callibration

The perspective projection discussed in section 2.2 is the starting point for mathematically modelling the imaging process. However, here there will be no simplifying assumptions about the relationship of the camera to the ground or about the ground geometry itself. The parameters y_c and f shown in Figure 1 are reduced to their most simple form $y_c = 0$, $f = 1$. However, where externally measured coordinates of points are available, these are in a separate coordinate system, called ground or object coordinates. The 3D coordinate system of Figure 1, having the center of projection at the origin and positive z-axis along the line of sight, is called the camera coordinate system. Coordinates (x', y') where a point is imaged on the plane $z = 1$ will be called virtual photo coordinates. We can think of these coordinates as measuring where a ray of light coming from an object into the camera intersects an imaginary glass pane placed one meter in front of (the center of projection of) the camera. The perspective projection (or collinearity) equations from section 2.2 reduce to

$$x' = x_c/z_c$$
$$y' = y_c/z_c$$

where (x_c, y_c, z_c) are the camera coordinates of a point. (The symbol y_c is being used differently here, the previous usage no longer being needed.) The parameters of the transformation from virtual photo coordinates to the pixel coordinates of the image are said to define the interior orientation (IO) of the photograph. If the camera is very well-behaved this transformation is simply a 2D scale and translation. However, we have found that lens distortion is too significant to ignore, so parameters for (radial) lens distortion are included as well, using a standard photogrammetric model. Currently it appears that these interior orientation parameters are consistent enough across various photos taken with the same camera, that we can treat these parameters as being camera constants, and we refer to determining these parameters as camera callibration. The interior orientation parameters are scale parameters c_x and c_y, translation parameters x_0'' and y_0'', and lens distortion parameters k_1, k_2, and k_3. The equations relating virtual photo to pixel coordinates are

$$x' = c_x(x'' - x_0'')(1 + k_1r + k_2r^2 + k_3r^3)$$
$$y' = c_y(y'' - y_0'')(1 + k_1r + k_2r^2 + k_3r^3)$$

$$\text{where } r = (x'' - x_0'')^2 + (y'' - y_0'')^2$$

The other parameters of the single photograph are six parameters of exterior orientation (EO), which relate object and camera coordinates; i.e. they specify how the camera was oriented relative to the external coordinate system when the picture was taken. The EO parameters are three angular parameters ω, ϕ, and κ and three translation parameters x_0, y_0, and z_0. The former are rotation angles about the three

coordinate axes and the latter are the object coordinates of the camera center of projection. Then camera coordinates (x_c, y_c, z_c) are related to object coordinates (x, y, z) by the following equations:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = M \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}, \text{where}$$

$$M = M_\omega M_\varphi M_\kappa = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega) & \sin(\omega) \\ 0 & -\sin(\omega) & \cos(\omega) \end{bmatrix} \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \begin{bmatrix} \cos(\kappa) & \sin(\kappa) & 0 \\ -\sin(\kappa) & \cos(\kappa) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The method used by Depth Finder to callibrate the camera is to combine the IO and EO parameters together into a single least squares adjustment. Two condition equations are derived by writing the original collinearity equations in the form

$$x_c - z_c x' = 0$$
$$y_c - z_c y' = 0$$

and then substituting the expressions obtained above for x_c, y_c, z_c, x', and y'. The adjustment seeks to find values for the 13 parameters (7 IO and 6 EO parameters) which best fit measurement data consisting of object and pixel coordinates for a set of points. Thus each measurement vector has 5 components (3 object coordinates and 2 pixel coordinates). The procedure assumes that independently obtained 3D object coordinates for the points are available; such points are called control points. The purpose of the adjustment is to obtain the IO parameters, but by carrying the EO parameters into the adjustment the control points can be in any convenient coordinate system without carefully positioning and orienting the camera relative to that system. Thus points may be marked and measured on some rigid object having a natural coordinate system of its own. As outlined in section 3.2, the adjustment procedure is iterative, and requires approximate initial values of the parameters.

3.4 Relative Orientation and Depth Computation

Given the IO parameters, which under our assumptions apply to both photos of a stereo pair, we now discuss the problem of computing the relative orientation of the two cameras relative to one another. Relative orientation may be regarded as a special case of exterior orientation in which the two coordinate systems are the left and right camera systems. In our conventions we treat the left camera as the one which is fixed, so that its coordinates are the object coordinates of EO. Thus the previous EO equations apply to RO by taking the camera coordinates (x_c, y_c, z_c) to be right camera coordinates (x_R, y_R, z_R) and object coordinates (x, y, z) to be left camera coordinates (x_L, y_L, z_L), giving

$$\begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} = M \begin{bmatrix} x_L - x_0 \\ y_L - y_0 \\ z_L - z_0 \end{bmatrix}$$

where the rotation matrix M is defined as before. A point has left and right virtual photo coordinates given by perspective projection as before:

$$(x'_L, y'_L) = (\frac{x_L}{z_L}, \frac{y_L}{z_L}), \quad (x'_R, y'_R) = (\frac{x_R}{z_R}, \frac{y_R}{z_R})$$

The left and right rays from perspective center to image point on each virtual photo are

$$\mathbf{r}_L = (x'_L, y'_L, 1)$$
$$\mathbf{r}_R = (x'_R, y'_R, 1)M$$

where $\mathbf{r}_R$ includes multiplication by M so that both rays are written in left camera coordinates. If these two rays are actually pointing at a common 3D point in the scene, then they are coplanar; the common plane also contains the baseline vector $\mathbf{b} = (x_0, y_0, z_0)$. This observation gives rise to the so-called coplanarity condition: $\mathbf{b} \cdot (\mathbf{r}_L \times \mathbf{r}_R) = 0$. We use this as the single condition equation in the least squares adjustment. In this adjustment a measurement vector has four components, consisting of the left and right pixel coordinates of a point measured on both photos of the stereo pair. The parameters to be determined are the six EO parameters x_0, y_0, z_0, ω, ϕ, and κ. However, each iteration of the adjustment is performed with x_0 fixed, so that only the five remaining parameters enter as variables in the function **f** discussed in section 3.2. This prevents the adjustment from reaching the trivial solution $\mathbf{b} = (0, 0, 0)$. After each iteration the baseline vector is rescaled to its initially given length, which is an input to the adjustment procedure. The standard photogrammetric procedure is to take this baseline length to be 1 during RO so that model coordinates are not scaled to real-world units, then to find the actual scale later during what is called absolute orientation. Our procedure at present is to take the baseline length as an input to orienting each stereo pair. Thus the user should measure or estimate the ground separation of the camera positions when the stereo photos are taken.

Once the relative orientation parameters have been computed, depth computations proceed by stereo reconstruction as follows. Given the left and right pixel coordinates, the IO parameters are used to compute left and right virtual photo coordinates. From these and the matrix M (computed using the RO angles), we can compute rays $\mathbf{r}_L$ and $\mathbf{r}_R$ from the equations above. Because measurement errors exist, these rays typically fail to exactly meet at the object point of interest, so we take the object point to be the midpoint of the line segment joining points of closest approach on the two rays. That segment is perpendicular to the two rays and hence its displacement is a multiple of $\mathbf{r}_L \times \mathbf{r}_R$, say $\gamma(\mathbf{r}_L \times \mathbf{r}_R)$. Letting $\alpha\mathbf{r}_L$ and $\beta\mathbf{r}_R$ be vectors from the left and right camera centers to the points of closest approach, it then follows that

$$\alpha\mathbf{r}_L + \gamma(\mathbf{r}_L \times \mathbf{r}_R) = \mathbf{b} + \beta\mathbf{r}_R .$$

The object point is $\alpha \mathbf{r}_L + (\gamma/2)(\mathbf{r}_L \times \mathbf{r}_R)$. To compute α and γ, take dot products of the equation above with $\mathbf{r}_L \times \mathbf{r}_R$ and $\mathbf{r}_R \times (\mathbf{r}_L \times \mathbf{r}_R)$, respectively to obtain

$$\gamma |\mathbf{r}_L \times \mathbf{r}_R|^2 = \mathbf{b} \cdot (\mathbf{r}_L \times \mathbf{r}_R)$$

$$\alpha |\mathbf{r}_L \times \mathbf{r}_R|^2 = (\mathbf{b} \times \mathbf{r}_R) \cdot (\mathbf{r}_L \times \mathbf{r}_R)$$

3.5 File Structure

In addition to the executable file DEPFIND.EXE, Depth Finder makes use of a number of files described below. Some of these must exist prior to running the program; others are generated when saving results.

(1) Initialization file

This file, named DEPFIND.INI, must reside in the same directory as the executable program file at startup. It is an ASCII text file holding initialization parameters. Each line is of the form

```
parameter_name = value
```

Here is an example of the file contents:

```
image_dir = c:\usr\images\
io = 359.160992 254.257195 0.001249 0.001249
lens = 0.003502 -0.000263 0.000016
selection_color = 255 130 125
eraser_size = 1
data_file = C:\usr\SED\depfind2\campus.txt
```

The first line above specifies the directory where image files are stored. The next two lines specify IO parameter values, the first line listing (in order) x_0'', y_0'', c_x and c_y and the second k_1, k_2, and k_3. The next two lines are the current settings for the selection color (red, green, blue) and eraser size in pixels. The last line specifies the data file, explained below, to be used at startup.

(2) Image files

Image files must reside in the directory specified in the initialization file and must at present be true color (24 bits per pixel) in the MicroSoft bitmap (BMP) format.

(3) Data files

One or more of these files may be used during a given program session. The data file loaded at startup is the one specified in the "data_file" line of the initialization file. A different file may be loaded

using the Open item on the File menu. The file contains sections labeled with the keywords within brackets; here is an example of the file contents:

```
[parameters]
left_image = cord01.bmp
right_image = cord02.bmp
base_length = 14.000000
linear_units = inches

[control]
   100 0.000000 0.000000 0.000000
   101 10.000000 0.000000 0.000000
   102 20.000000 0.000000 0.000000
   103 30.000000 0.000000 0.000000
   104 40.000000 0.000000 0.000000
   105 50.000000 0.000000 0.000000

[measurements]
image = cord01.bmp
  100 44736 36352
  101 66368 35712
  102 88128 35136
  103 109888 34688
  104 131200 34176
image = cord02.bmp
  100 32512 35904
  101 53248 35136
  102 74624 34624
  103 96320 33856
  104 118080 33280
  105 139776 32832
```

The parameters section contains the filenames of the left and right image files, the base length, and a setting for linear units, which can be "meters" or "inches." The base length and coordinates of control points in the data file, as well as the display of other coordinates in dialog boxes, are given in linear units. The control section gives 3D object coordinates of control points; the first field is an integer "point id" identifying the point. The measurements section is divided into subsections headed by an "image = filename" line specifying an image, followed by point measurements for that image. The measurement lines consist of a point id followed by subpixel x and y coordinates, where a subpixel is 1/256 of a pixel.

(3) Depth file

The output of the program is a depth file corresponding to the left image of the stereo pair, containing a 16-bit depth value for each pixel of that left image. The filename is derived from the image filename by changing the extension from "BMP" to "RNG". The file has no header or other padding, just the depth values starting with the upper left pixel and proceeding by rows down the image. The depth values are interpreted as

0 = unassigned depth

1 = sky

2 - 65535 = unsigned integer depth in tenths of a meter

3.6 Program Operation

Because Depth Finder is still under development the material in this section is likely to undergo significant changes. For that reason, these instructions will not be as detailed as those given for PathEdit. At startup Depth Finder reads the initialization file and if a data file is specified there, reads that file as well. This tells the program what images to load. At any given time the program can have one or two images loaded. When two images are loaded the main program window is split into two side-by-side image display windows as shown in Figure 5. Any point measurements found in the data file are indicated by cross symbols. The display of these symbols can be suppressed by unchecking that option under the Options menu. A different data file may be opened using the Open menu item. The current point measurements as well as other data may be saved into the most recently opened data file with the Save menu item, or saved to a new data file using Save As. Different images may be loaded without reading a different data file using the Select Images menu item. The Close Right Image item allows the user to switch from having two images loaded to just one.

Various program operations cause messages to be written into the message log, which can be displayed using the Message Log item on the Dialogs menu. The Write Message Log item under the File menu causes the current contents of the message log to be written to a file named msglog.txt.

To measure points, first display the Point List dialog box (under the Dialogs menu) shown in Figure 6. The active point, whose id is shown in the field near the top, is the point currently being measured. If that point already has a measurement on one or both photos, its measured coordinates are shown in the point list below, and its symbol changes from red to yellow in the image display. The active point may be changed in several ways: (1) selecting a row in the point list or control point list, (2) keying in a new point id and hitting the apply button, (3) hitting the plus button, or (4) using the grab point command. The plus button causes the point id of the active point to increment to the next larger integer; it is convenient for measuring several points with sequential point ids. The grab point command is activated under the Commands menu; once activated if the left mouse button is hit with the mouse cursor in the image display window near a measured point, the closest such point becomes active. If Active Point Centering (under the Options menu) is turned on, then each time a previously measured point becomes active the image or images on which it is measured scroll so as to center the measured point in the display window. Once a point is active and the Measure Point command is active (under the Commands menu), hitting the left mouse button with the mouse cursor in the image display window causes the current mouse cursor position to be interpreted as the point's measurement. If it was previously measured, that measurement is updated. The coordinates are displayed in the point list. Typically in measuring a point the user will want to scroll the image or images to the area of interest, then use the Zoom In command to allow precise positioning of the mouse cursor.

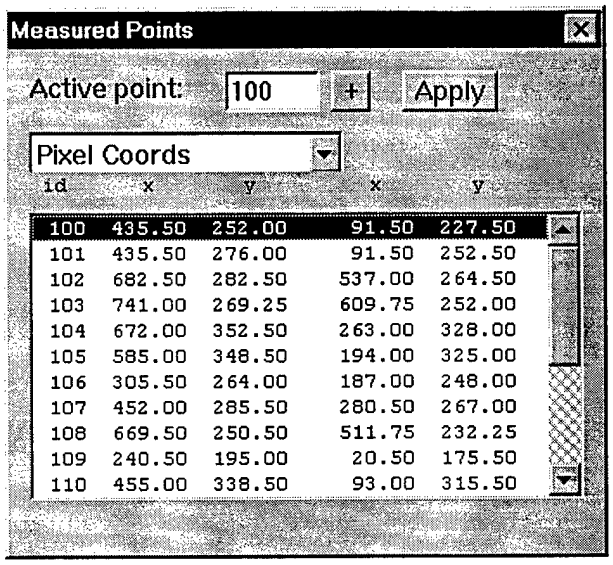


Figure 6

To perform camera callibration, a collection of control points must be externally measured and those measurements entered into a data file (under the control section). A single photo must be taken so that the control points appear in a well-distributed manner (extending as much as possible across the image from top to bottom and right to left). The points should then be measured on this photo, using the point ids given to the points in entering their control coordinates in the data file. This may be conveniently accomplished by displaying the Control Point dialog box (under the Dialogs menu) and selecting the points listed there in turn to make them active. After the measurements are completed, select Camera Callibration on the Commands menu. Results are displayed on the message log. Also adjusted point symbols may be displayed along with the measured point symbols, using Show Adjusted Points on the Options menu. Select Save on the File menu to save the measurements.

To perform relative orientation, choose a data file designating left and right images and a baseline length. Measure at least six points on the two images, then select Relative Orientation on the Commands menu. Results are displayed on the message log. Also the point list can be switched to show model coordinates rather than pixel coordinates. These are the result of the stereo reconstruction process described in section 3.4, with the z-coordinate representing depth.

To begin creating or editting the depth image, first display the Range Dialog (from the Dialogs menu), which is shown in Figure 7. Note that the term "range" is used interchangeably with "depth" here. The range image is based on the left image, so it may be helpful to select Left Only under Images To Display on the View menu. Next define a region within which depths will be assigned; this region is called the current selection. The Select menu contains items for selecting individual pixels, lines, or rectangles in the manner of a paint program. The current selection is highlighted over the left image using the current selection color, which can be changed by choosing "Set Selection Color ...".

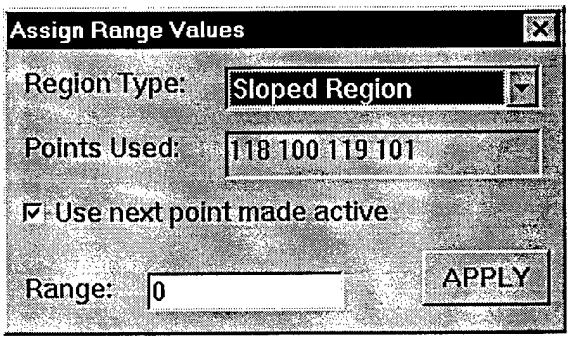


Figure 7: Range Dialog

The Range Dialog allows for three region types: Sky, Single Range Region, or Sloped Region. Actually Sky and Single Range Region are essentially the same, each being a region in which a single range value will be assigned to every pixel; the only difference is that the special range value 1 is used for Sky, whereas the range for a Single Range Region is based on that of a measured point. To select the point whose range is to be used, make sure there is a check in the box labeled "Use next point made active"; then make the desired point the active point as discussed above. The most natural way to do this is probably by using the grab point command (Commands menu). The point id will appear next to Points Used on the Range Dialog. It is also possible to type a range value into the field labeled Range. At this point, hitting the Apply button will cause the range value to be assigned to the current selection. To assign ranges to a sloped region, at least three points must be used. Points may be added to the list of points used by making them active. The Points Used field may be cleared by reselecting a region type. Again once the desired points have been selected, hit the Apply button to assign ranges to the current selection. This time the ranges are assigned based on the plane which best fits the points used. To view the resulting range assignments, use Clear Selection on the Select menu and then Overlay Range on the View menu. This displays ranges in a pseudocolor manner over the left image.

The exact order of the operations described in the previous paragraph is not important. For example, the current selection may be defined after choosing points for a sloped region, or part of the region selected before choosing points and part after. Also additional points to be used may be measured during the assignment of ranges; display the point list dialog box and proceed to measure new points as

described earlier. Note however that a point must be measured on both images for its model coordinates to be computable, giving it a range.

Pixels may be removed from the current selection by "erasing" them out of the selection using the right mouse button, while Mark Pixels on the Select menu is checked. Eraser size options are provided under the Options menu analogous to brush size in a paint program.

A method is provided for selecting groups of pixels having similar color values. To do this first select a region to be used as a sample. Then go to Record Sample Statistics on the Select menu. The minimum and maximum values for each of red, green, and blue are computed. Now select a larger region containing the region of interest and finally go to Select By Sample on the Select menu. The current selection is trimmed back to include only those pixels falling within the ranges for red, green, and blue found earlier.

If the current selection includes pixels to which ranges have already been assigned, the new assignment will override the previous assignment. If the user wishes to avoid this happening, the Clear Pixels with Ranges item on the Select menu may be used to remove such pixels from the current selection. On the other hand, overriding previous assignments is useful in assigning depths first to ground pixels and then to occluding objects above the ground.

The Save item on the File menu saves both point measurements into the data file and the current range image. The Save As item only allows for renaming the data file; the filename for the range image is always based on the image filename as discussed in the previous section.